Amendments to the Claims:

Please amend claim 1, as shown below.

1. (Currently Amended): A system, comprising:

one or more compilers, executed by one or more processors, wherein the compilers support mixing and nesting of languages within a source file;

an extensible multi-language compiler framework, wherein the compiler framework provides a language-independent source code editor with information about the source file, comprising[[:]] signatures of classes defined by the source file, errors found in the source file, stack of nested languages at any point in the source file, and information exposed by any languages; ~~and~~

the language-independent source code editor, wherein the language-independent source code editor communicates to the compiler framework using language-independent metadata;

wherein the extensible multi-language compiler framework has error correction in code-generation, permitting a user to run code even if there is an error in the code;

wherein a thread pool allows compilation of multiple files to be performed in parallel; and

wherein a type cache contains signatures for classes.

2 – 5. (Canceled)

6. (Previously Presented): The system of claim 1, wherein integrating a new language does not require separate instructions to enable compiling or editing of the new language.

7. (Previously Presented): The system of claim 1, wherein the language-independent source code editor displays errors for mismatched start and end XML tags embedded in the source code and performs auto-completion of XML tags embedded in the source code.

8. (Previously Presented): The system of claim 7, wherein the language-independent source code editor displays errors and performs auto-completion independent of the host language embedding XML tags.

9. (Previously Presented): The system of claim 1, wherein the language-independent source code editor provides syntax coloring and code completion for editing annotations.

10 (Previously Presented): The system of claim 1, wherein the compiler framework makes it possible to reparse in near real-time with no performance degradation noticeable to the user.

11. (Previously Presented): The system of claim 1, wherein the compiler framework enables the language-independent source code editor to provide visual indication of errors throughout a source file with mixed languages.

12. (Previously Presented): The system of claim 1, wherein the compiler framework keeps track of errors in source files in a project so that a user can have a list of errors in opened and unopened source code files in a project.

13. (Cancelled)

14. (Previously Presented): The system of claim 1, wherein the compiler framework allows an outer language compiler to pass off processing of a section of a document to an inner language compiler.

15. (Previously Presented): The system of claim 14, wherein a parse tree produced by the inner compiler is available to the outer compiler.

16. (Previously Presented): The system of claim 15, wherein either the inner compiler or the outer compiler can determine where the span of the inner compiler's language content ends.

17. (Previously Presented): The system of claim 1, wherein the compiler framework includes a parser generator and a scanner generator.

18. (Previously Presented): The system of claim 17, wherein generated parsers are able to recover from all single token errors and missing identifiers that occur during code completion.

19. (Previously Presented): The system of claim 1, wherein the compiler framework provides the source code editor with names of classes and packages in a project and errors found in any source files in a project.

20. (Previously Presented): The system of claim 19, wherein after the compiler framework is notified of a change to a file, the information about the file is updated within a time limit for a single-file recompile.

21. (Previously Presented): The system of claim 20, wherein after the file is recompiled, the compiler framework provides the source code editor with a list of changes that occurred to the file information.

22. (Previously Presented): The system of claim 1, wherein the compiler framework includes a project compiler, wherein the project compiler contains a list of source directories and the class path and maintains a type cache which contains signatures for classes in a project.

23. (Previously Presented): The system of claim 22, wherein the type cache is indexed by file name and by class name, and maintains a current list of errors and a list of dependencies.

24. (Previously Presented): The system of claim 23, wherein the project compiler and the type cache are serializable.

25. (Previously Presented): The system of claim 1, wherein a file compiler is used to perform compilation of a single source file.

26. (Previously Presented): The system of claim 25, wherein the file compiler supports interoperation of different languages by using a common intermediate language.

27. (Cancelled)

28. (Previously Presented): The system of claim 25, wherein the file compiler remembers where the language nesting occurs for reuse on subsequent parses.

29. (Previously Presented): The system of claim 28, wherein the outer language implements a name resolution interface to allow the inner language to resolve references to names defined outside of the nested language.

30. (Previously Presented): The system of claim 1, wherein all parsing is performed on background threads.

31. (Cancelled)